

NintendoWare for Cafe

Sound Development Environment Overview

2013/10/02

Version 1.0.1

**The content of this document is highly confidential
and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and/or its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Introduction	5
1.1	Introduction to NintendoWare Sound Development Environment	5
1.2	Features of the NintendoWare Sound Development Environment.....	5
1.2.1	Sound Archives	5
1.2.2	GUI Tools (SoundMaker).....	5
1.2.3	Development Environment for Sounds Alone	5
1.2.4	Data Compatibility	6
1.2.5	Changes from NintendoWare for Revolution	6
2	Folder Structure	9
3	Sound Development Environment Configuration	10
3.1	Flow of Sound Development.....	10
3.1.1	Creating Data	11
3.1.2	nw::snd Library Features.....	11
4	Main Sound Representations in NintendoWare.....	12
4.1	Supported Replay Formats	12
4.1.1	Stream Sounds.....	12
4.1.2	Sequence Sounds	12
4.1.3	Wave Sounds	12
4.2	3D Sounds	12
5	Sound Runtime Library	13
5.1	nw::snd Library.....	13
5.2	nw::snd Library Configuration	13
5.2.1	Sound Archive Player	14
5.2.2	Sound Archives	14
5.2.3	Sound Handles.....	14
5.3	Memory Management of Sounds.....	14
5.3.1	Sound Heap	14
5.3.2	Groups.....	15
6	GUI Tools	16
6.1	Sound Data Development Using SoundMaker	16
6.1.1	Linking with Existing Applications.....	17
6.1.2	Creating Sound Archives.....	17
6.1.3	Linking Using Multiple Staff Members.....	17
6.1.4	Checking Each Sound.....	18

Figures

Figure 2-1 NintendoWare Folder Hierarchy	9
Figure 3-1 Flow of Sound Development.....	10
Figure 5-1 nw::snd Library Configuration	13
Figure 5-2 Allocating and Freeing the Sound Heap	15
Figure 5-3 Loading Groups	15
Figure 6-1 SoundMaker Operational Concept 1	16
Figure 6-2 Example of Collaboration Using SoundMaker	18
Figure 6-3 SoundMaker Operational Concept 2	18

1 Introduction

This document describes the sound development environment provided by NintendoWare for Cafe (NintendoWare).

1.1 Introduction to NintendoWare Sound Development Environment

The NintendoWare sound development environment is used to carry out sound development for Cafe game software. It consists of the runtime library (nw::snd library), NW4F SoundMaker (which is a dedicated GUI tool for creating sound data for nw::snd library), and command-line tools for conversion of various sound data types.

1.2 Features of the NintendoWare Sound Development Environment

The NintendoWare sound development environment was created by combining elements of the development environments used for the Nintendo GameCube™, Nintendo DS, Wii, and Nintendo 3DS.

This section describes the main features of the NintendoWare sound development environment.

1.2.1 Sound Archives

Resource data used by nw::snd library are combined into a single collection of data called a *sound archive*.

Since gathering all the wave data used in streaming would result in a very large archive, such data is managed at the wave file level, instead of including it in the sound archive.

1.2.2 GUI Tools (SoundMaker)

The NintendoWare sound development environment provides SoundMaker, a dedicated tool to support the entire process of creating sound data.

SoundMaker is used to check sounds on a console connected to a development PC and SoundPlayer compatible with HostIO.

You can also check some sounds during their creation, using emulation on a PC.

1.2.3 Development Environment for Sounds Alone

The nw::snd library of the NintendoWare sound development environment uses as its base engine the AX library, which is the standard audio engine of the Cafe SDK. If you plan to use only the sound development environment of the NintendoWare development environment, extract the nw::snd library and the system libraries (the nw::ut library and other system libraries).

1.2.4 Data Compatibility

In the NintendoWare sound development environment, data in the format of WAV files and AIFF files can be used for wave data, and data in the format of standard MIDI files can be used to sequence data (musical score information).

NintendoWare also supports the importing of data from these past sound development environments:

- NITRO-Composer (the Nintendo DS/Nintendo DSi sound development environment).
- NintendoWare for Revolution (the Wii development environment).
- NintendoWare for CTR (the Nintendo 3DS development environment).

1.2.5 Changes from NintendoWare for Revolution

The following changes have been made since the NintendoWare for Revolution sound tools.

1.2.5.1 (Converter) Support for Parallel Conversion

You can now convert sounds in parallel if they do not depend on each other. The number of sounds that can be converted simultaneously is configurable from the **Options** screen.

1.2.5.2 (Converter) Ability to Suppress String Table Output

You can now suppress string table output. Although NW4R SoundMaker forced string tables to be output to sound archives, you can now prevent this by opening the **Project Settings** and clearing the **Output String Tables** check box on the **Sound Archive** tab. This allows you to reduce the size of sound archives when sounds are not called using strings.

1.2.5.3 (Converter) Specification Change for Sound Archive Map Files

The conversion process now generates HTML map files, rather than text files, for sound archives.

1.2.5.4 (Converter) IDs Given to Wave Sound Sets and Sequence Sound Sets

Item IDs are now given to wave sound sets and sequence sound sets, allowing you to specify these items when you load data.

1.2.5.5 (Converter) Specification Change for Item IDs

Information that shows an item's category is now embedded in the upper 8 bits of each ID for the various sound types (sequence sounds, wave sounds, and stream sounds), sequence sound sets, wave sound sets, banks, wave archives, groups, and players.

1.2.5.6 (Converter) Specification Change Related to Cache Output

Intermediate binary files are now always output to the cache folder. However, bank instrument list files (`.finl`) and intermediate text sequence sounds (`.fseq`) generated by SMF are still output to the same location as the source files.

1.2.5.7 (SoundMaker) Specification Change for Preview Playback

The specifications for playing back previews of sounds and instruments have been reviewed. The following changes were made to playback of sound previews.

- The spacebar registers a sound with the preview player and starts playback.
- Double-clicking the list does not cause a sound to be registered and played back with the preview player.

The following changes were made to playback of instrument previews.

- Pressing F9 or F10, or double-clicking the list, does nothing.
- Hold playback only occurs while the spacebar is held down.

1.2.5.8 (SoundMaker) Wave Archive Nodes Added to the Project Tree

Wave archive nodes have been added to the project tree. You can double-click on a wave archive node to display the wave archive tab.

1.2.5.9 (SoundMaker) Wave Archive List Added

A wave archive list has been added to the sound set tab. To display it, double-click a wave archive node in the project tree. The wave archive list shows the following three items: **Name**, **Separate Loading**, and **Comments**.

1.2.5.10 (SoundMaker) Dialog Box Added for Creating a New Wave Archive

A dialog box has been added for creating a new wave archive. You can select **Add Item** or **Insert Item** from the shortcut menu in the wave archive list to display a dialog box that allows you to choose **Name** and **Separate Loading**.

1.2.5.11 (SoundMaker) Output Format Column Added to the Group List

The group list's **Output Format** and **Wave Sharing** columns have been deleted and the **Output Format** column added. You can set the output format to **Link**, **Embed**, or **No Output**.

1.2.5.12 (SoundMaker) Multi-Bank Dialog Box Added to the Sequence List

You can select **Multi-bank Settings** from the sequence list's shortcut menu to display a multi-bank dialog box. You can specify up to four banks.

1.2.5.13 (SoundMaker) Preview Bank Window Added

The Preview Bank window has been added. To display it, select **Preview Bank** from the **View** menu. Input MIDI signals from a MIDI sequencer are played back using the bank registered in this window.

1.2.5.14 (SoundMaker) Specification Change Related to the Target Bank for MIDI Keyboard Input

The active bank panel is now the target bank for MIDI keyboard input. No audio is played if there are no active bank panels. Buttons to display the bank and instruments were removed from the toolbar together with this change.

1.2.5.15 (SoundMaker) Specification Change Related to MIDI Settings

You can now choose from MIDI keyboard input and MIDI sequencer input. The active bank panel's bank is used to play back MIDI signals from the device configured for MIDI keyboard input. The

Preview Bank window's bank is used to play back MIDI signals from the device configured for MIDI sequencer input.

1.2.5.16 (SoundMaker) Import of NW4R SoundMaker Projects

You can import the following files from NW4R SoundMaker projects: project (`.rspj`) files, sound-set (`.rsst`) files, bank (`.rbnk`) files, and text-sequence sound (`.rseq`) files. The original files are not changed when you import them; new files are saved using a different extension in the same folder.

Text-sequence sound files are imported as follows.

- A batch replace is used to change the strings `“rseq”` and `“rinl”` to `“fseq”` and `“finl”` within the file.
- The `“rseq”` files are imported recursively.

2 Folder Structure

The following figure shows the folder hierarchy of sound-related tools and documents provided by NintendoWare.

Figure 2-1 NintendoWare Folder Hierarchy

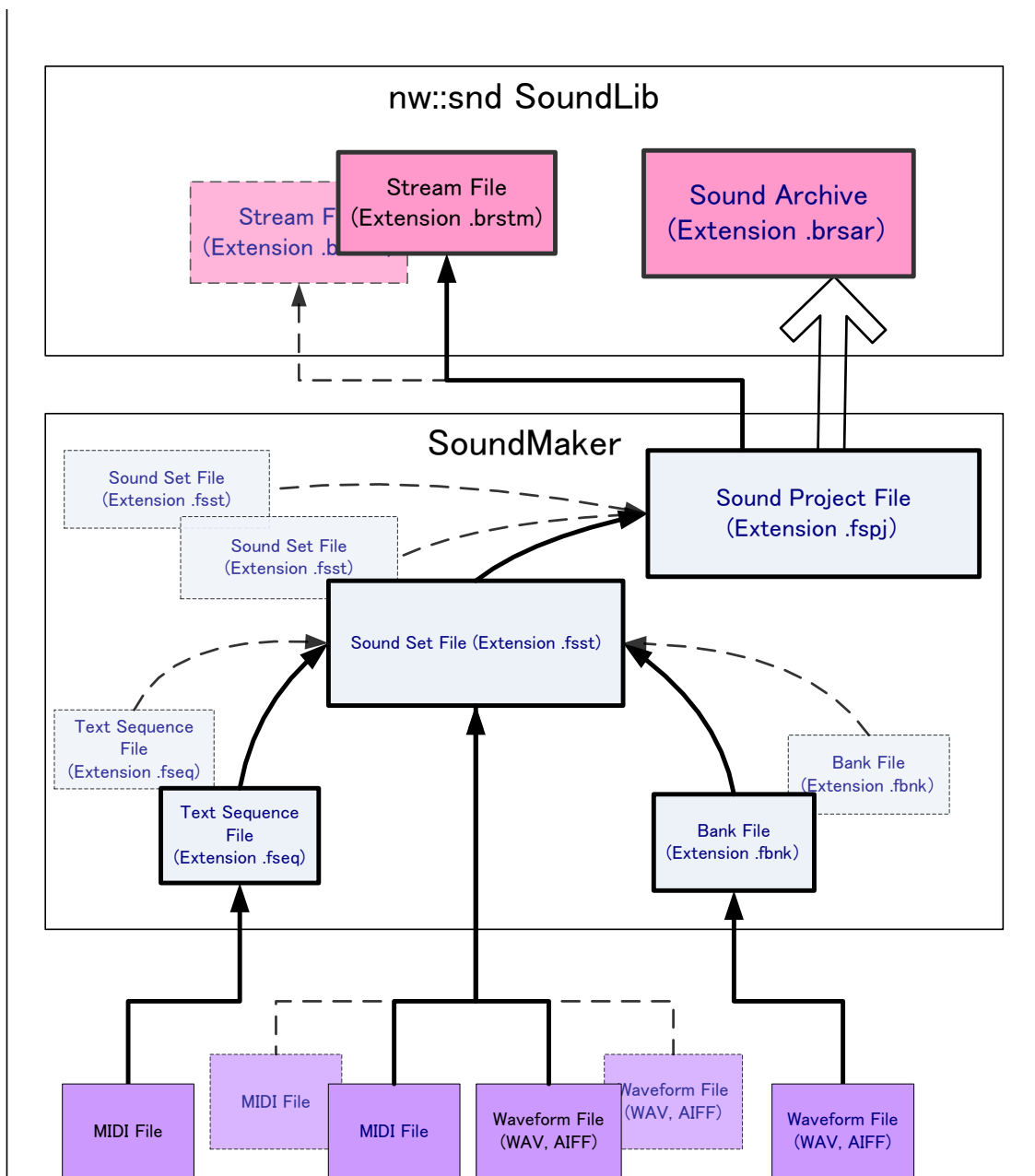
```

NW_Cafe (The environment variable NW4F_ROOT points to here)
├ Demo
│   ├── discdata // Folder for mounting hardware
│   │   └ snd // Sound library demo data
├ Document // Documentation
│   ├── API // Function reference
│   ├── Overview
│   │   └ Sound
│   │       └ Sound_Overview.pdf // Overview of sound development environment
│   │           (this document)
├ ReleaseNotes // Release notes
│   └ Changelogs
│       └ Sound_ChangeLog.html // Sound revision history
└ Sound
    ├── QuickStart
    │   └ index.html // Sound Quick Start Guide
    ├── Sound_ProgrammersGuide.pdf // Sound Programmer's Guide
    ├── Sound_SequenceDataManual.pdf // Sound Sequence Data Manual
    ├── SoundProjectConverterManual.pdf // Sound Project Converter Manual
    └ SoundMaker
        └ SoundMaker.html // SoundMaker User's Guide
Library // Libraries
└ Snd // The nw::snd library
SampleData // Various sample data
└ Sound // Sample data for SoundMaker
Tool // NW4F Tools
└ SoundMaker
    ├── NW4F_SoundMaker.exe // SoundMaker
    ├── NW4F_SequenceConverter.exe // Sequence Converter
    ├── NW4F_SmfConverter.exe // smf Converter (Used in SoundMaker)
    ├── NW4F_SoundProjectConverter.exe // Sound Project Converter
    └ MusTableAnalyzer.exe // mus Importer (Used in SoundMaker)
└ SoundPlayer // SoundPlayer
  
```

3 Sound Development Environment Configuration

3.1 Flow of Sound Development

Figure 3-1 Flow of Sound Development



3.1.1 Creating Data

MIDI sequence data and wave data are created using commercially available sequencers and wave-editing software, respectively. SoundMaker uses a runtime library to create a usable sound archive from the standard MIDI files (SMF files) and wave files (AIFF and WAV files) output by such software.

3.1.2 nw::snd Library Features

The nw::snd library uses sound archives to play and control each sound (for example, to change volume and change pitch) and to perform memory management.

The nw::snd library can be used to play three types of sound files: stream sounds, sequence sounds, and wave sounds.

Sound control uses a mechanism known as a *sound handle* to seamlessly change parameters like volume and pitch of sounds that have different replay formats. At the same time, each sound is managed by a player. The number of voices to be played and the priority of voices are controlled at the player level.

The nw::snd library implements an original method of memory management called a *sound heap*. The various types of sound data can be loaded into memory separately, or loaded in units of *groups* as set by the sound designer.

4 Main Sound Representations in NintendoWare

4.1 Supported Replay Formats

4.1.1 Stream Sounds

Stream sounds are played back as successive parts of the wave data are loaded into memory. It is also possible to play back more than one stream sound at once. Playback is possible using only the minimum amount of memory required, even if the wave data is long, because the entire data does not have to be located in memory.

Stream sounds are suitable for playing extended wave data, such as BGM or other sounds.

4.1.2 Sequence Sounds

Sequences can be played back using sequence data and sound source data. Data converted from a standard MIDI file can be used as sequence data. Properties such as tempo and volume by track can be freely changed when using sequence playback. You can also keep the data size required relatively small.

Sequence sounds are suitable for playback of BGM as well as scale-like sound effects.

4.1.3 Wave Sounds

Although wave data is played in order to replay wave sounds as with stream sounds, wave data must be loaded into memory ahead of time. This reduces the load, compared to stream sounds, when performing repetitive playback because there is no need to load data each time.

Wave sounds are suitable for the playback of relatively small amounts of wave data, such as sound effects.

4.2 3D Sounds

3D sound can create spatial representations such as distance and angle by using sound emissions based on the direction and coordinates of the *3D sound listener* (a microphone in the real world), coordinates of the *3D sound actor* (a sound source in the real world), and the sound request made for the 3D sound actor.

3D sound is ideal for representing sounds in games with characters that move around freely in 3D space.

5 Sound Runtime Library

5.1 nw::snd Library

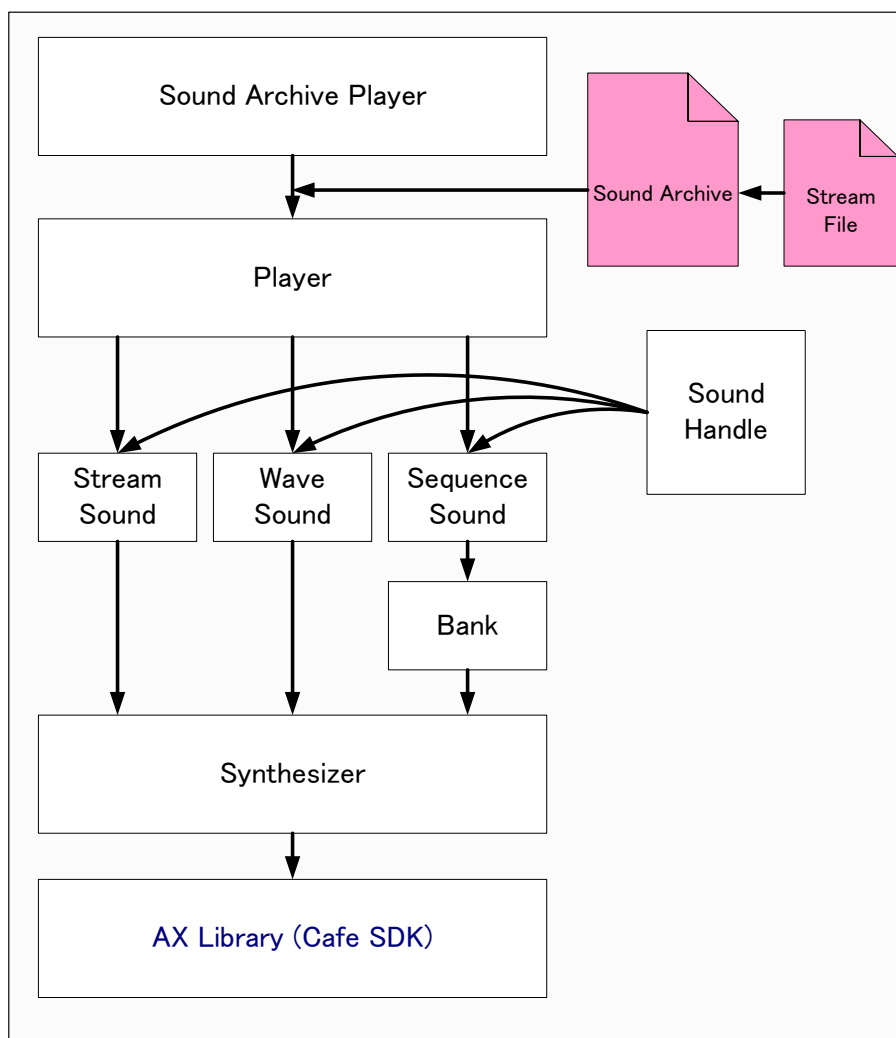
The nw::snd library is designed to replay sound data created for the sound runtime library on the CTR system.

The nw::snd library is a C++ class library. Each sound feature is implemented by class.

5.2 nw::snd Library Configuration

The following diagram shows the configuration of the nw::snd library.

Figure 5-1 nw::snd Library Configuration



5.2.1 Sound Archive Player

The sound archive player class is central to the `nw::snd` library. Programmers normally play sounds using a sound archive player function.

A sound archive player can play back a sound archive (a collection of sound data) using a player to replay the sounds.

5.2.2 Sound Archives

A sound archive class corresponds to the data in a single sound archive (`bfsar` file). Sound archives are used to handle data input and data analysis of sound archive data located in storage.

5.2.3 Sound Handles

Various forms of playback control using handles are available for sounds played on a player, including pausing and changing the audio volume.

Since the same interface is used for sound handles, whether the sound being controlled is a sequence sound or stream sound, you do not need to be aware of differences in the types of sounds.

Another handle, called a *sequence sound handle*, is also provided to control sequence sounds and support operations, such as changing tempo, and can only be employed when using sequence sounds.

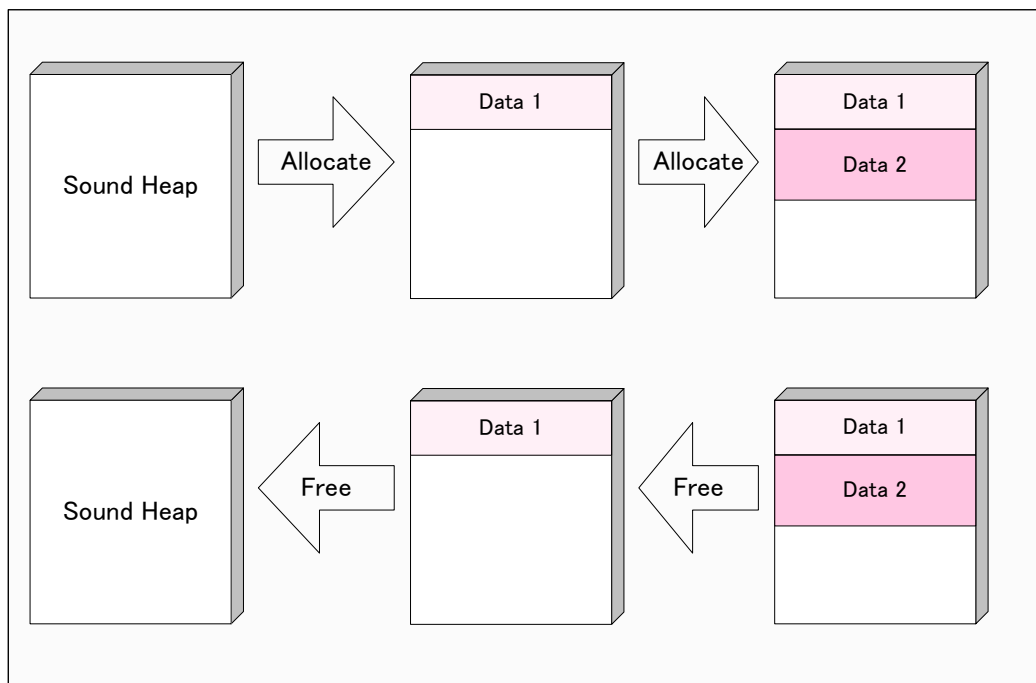
5.3 Memory Management of Sounds

Memory management of sound data is provided using a sound heap, which is an easy way to use a heap system that sound designers can use.

5.3.1 Sound Heap

A sound heap is used to store sound data in a sound archive.

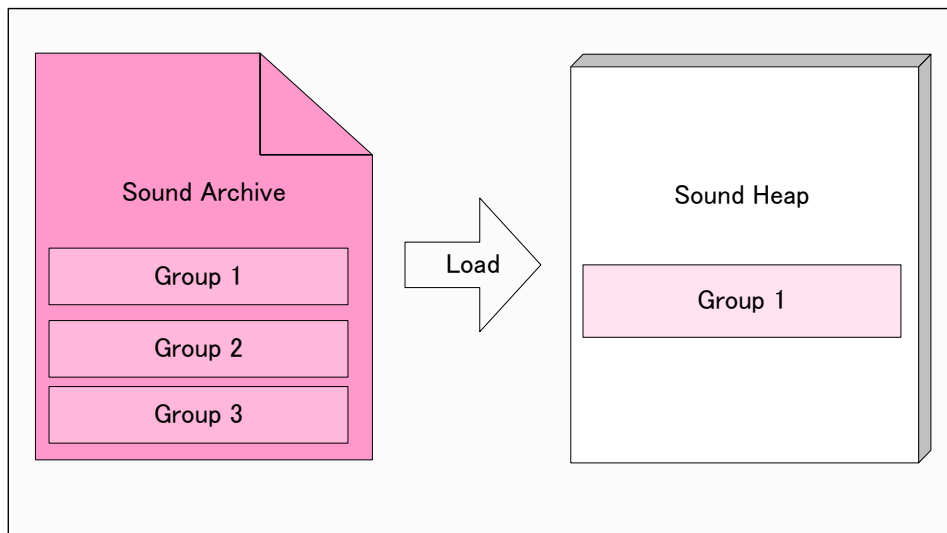
Using a heap system, memory is allocated in order from the top of the sound heap. When deleting unneeded data, it is freed in the order of most recently allocated.

Figure 5-2 Allocating and Freeing the Sound Heap

5.3.2 Groups

When sound data are loaded to the sound heap, the sets of data can be loaded separately, or collected together and loaded in units called *groups*. This section explains sound data stored as groups in the sound archive.

Sound designers separate multiple sets of sound data into groups each time data is loaded. Programmers load sound data into the sound heap at the group level.

Figure 5-3 Loading Groups

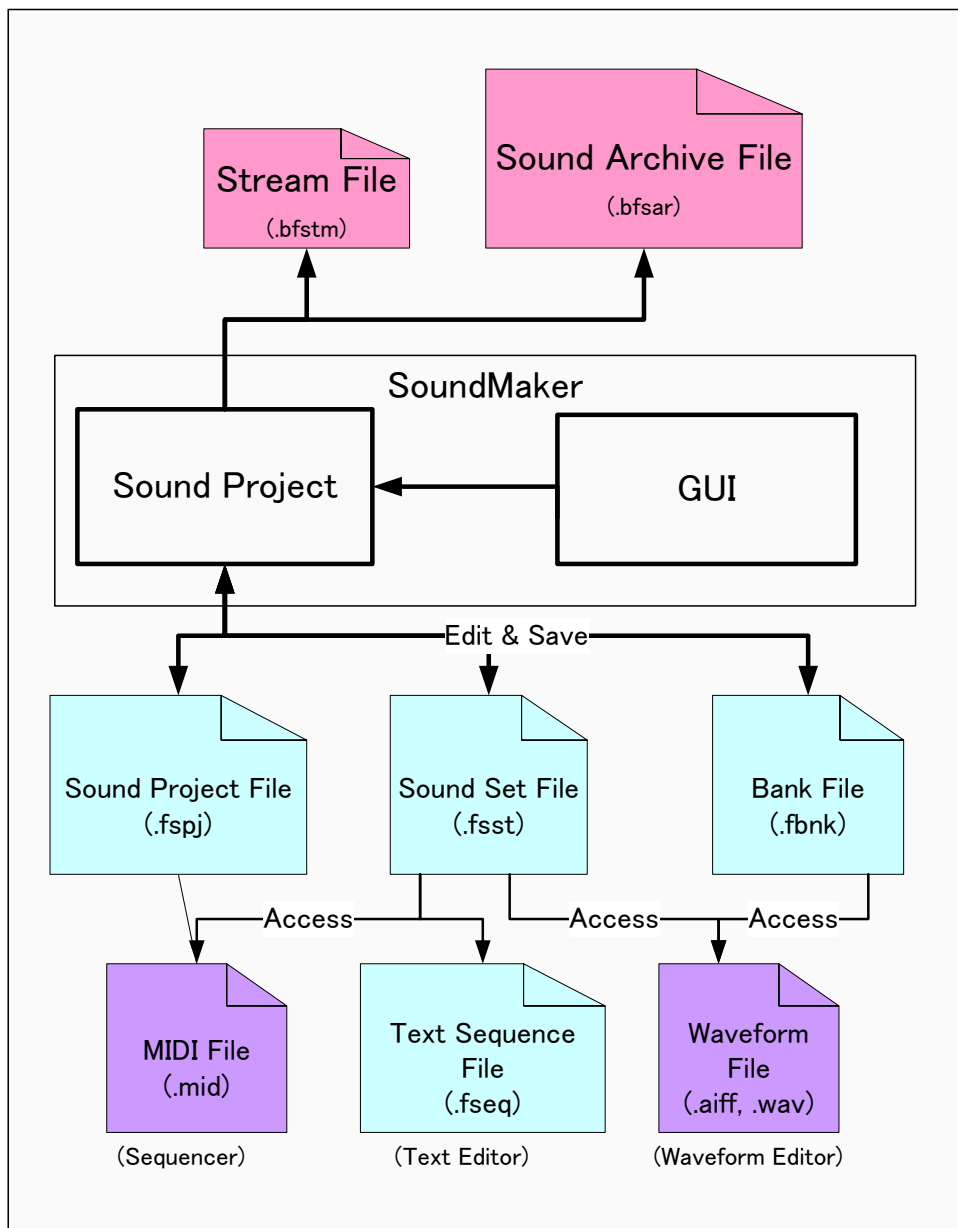
6 GUI Tools

6.1 Sound Data Development Using SoundMaker

SoundMaker is a special NintendoWare GUI tool that runs on Windows.

The following diagram shows the flow of data when using SoundMaker.

Figure 6-1 SoundMaker Operational Concept 1



6.1.1 Linking with Existing Applications

There are many excellent existing applications for sound development available, with each sound designer using his or her favorite, depending on the application. SoundMaker emphasizes linking with these frequently used, existing applications.

For wave data, the supported formats files are WAV files (RIFF files), most commonly used on Windows, and AIFF files, commonly used on the Mac OS. These wave files are used in the creation of stream sounds, sequence sound banks, and wave sounds.

For sequence data (musical score information), the supported format is Standard MIDI files, which are supported by nearly all sequencers.

With SoundMaker, you can use this wave data and sequence data to create data specialized for Cafe applications.

6.1.2 Creating Sound Archives

Sound projects are project files created using SoundMaker. One sound project corresponds to one sound archive (*bsfar* file).

Multiple small-scale collections of sounds called *sound sets* can be registered in the sound project.

Various settings, such as **Number of sequence sounds**, can be configured for the sound project. This information affects the size of work memory required by the `nw: :snd` library.

The actual creation of sounds is carried out using sound sets.

The sound set accesses the files required for each sound: MIDI files, wave files, bank files used as sound sources by MIDI files, and the text sequence file in which sequence commands have been coded.

The sound archive is created from all of the configured information and registered files in the sound project.

6.1.3 Linking Using Multiple Staff Members

All sounds are saved in the sound sets (sound set files) accessed by a sound project, including the following:

- Stream sounds
- Wave sounds
- Sequence sounds
- Information regarding the specified player to be used by each sounds

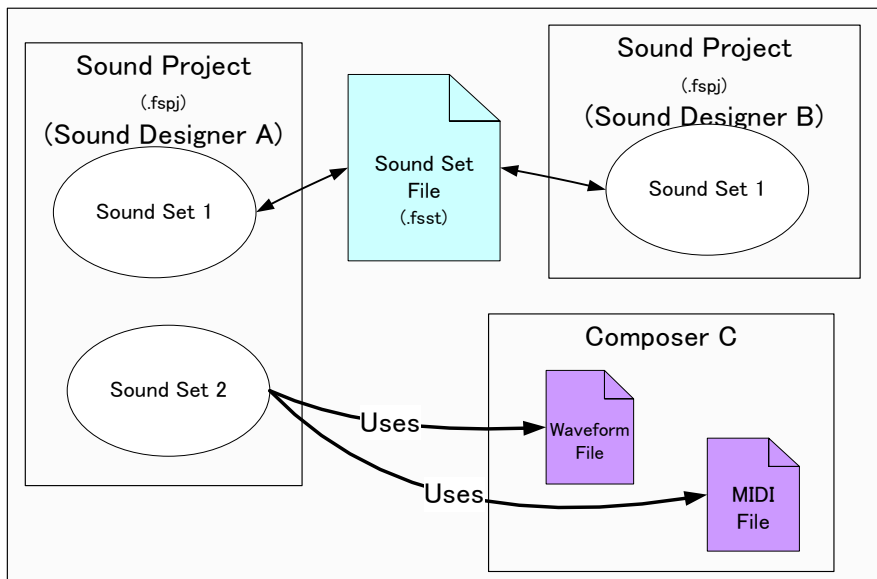
Nearly all this information can be used to build the sound archive, including which banks are accessed by sequences.

In addition to the ability to share data and pass it back and forth using wave files and MIDI files, data can be shared more efficiently when a staff is working cooperatively using SoundMaker, dividing

responsibilities and work at the sound set file level.

For details, see the figure below.

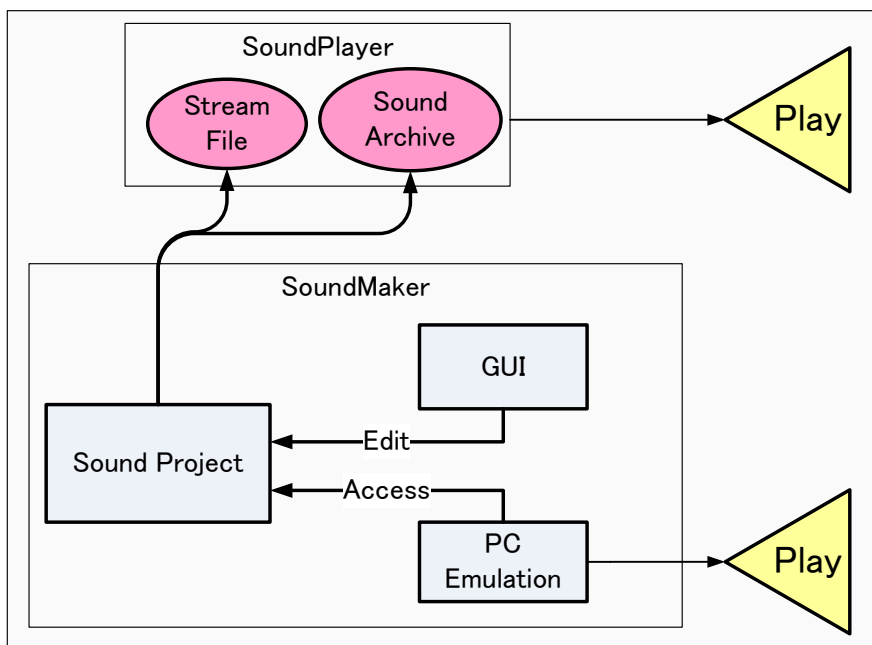
Figure 6-2 Example of Collaboration Using SoundMaker



6.1.4 Checking Each Sound

While creating sounds with SoundMaker you can check them via either SoundMaker's own PC emulation playback or by playback on the actual hardware using SoundMaker..

Figure 6-3 SoundMaker Operational Concept 2



6.1.4.1 Playback on the Actual Hardware

Pass the converted sound archive to SoundPlayer and start, then operate the controller to play the created sound data on the actual hardware.

6.1.4.2 Playback Using PC Emulation

SoundMaker can be used to check sounds while they are being created using PC emulation.

PC emulation supports MIDI and can be used to play instruments from MIDI keyboards and sequencers connected to the PC.

Revision History

Version	Revision Date	Description
1.0.1	2013/10/02	Deleted Section 1.2.6 Changes from NintendoWare for CTR.
1.0.0	2011/07/06	Initial version.

All company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2011–2013 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.